CLAIMS

What is claimed is:

1. An asymmetric data processor comprising:

one or more host computers, each including a memory, a network interface and at least one CPU, each host computer being responsive to requests from end users and applications to process data;

one or more Job Processing Units (JPUs), each having a memory, a network interface, one or more storage devices, and at least one CPU, each JPU being responsive to requests from host computers and from other JPUs to process data;

a network enabling the host computers and the JPUs to communicate between and amongst each other, each of the host computers and JPUs forming a respective node on the network; and

a plurality of software operators that allow each node to process data in a record- by-record, streaming fashion in which (i) for each operator in a given sequence of operators, output of the operator is input to a respective succeeding operator in a manner free of necessarily materializing data, and (ii) data processing follows a logical data flow and is based on readiness of a record, such that as soon as a subject record is ready record data is passed for processing from one part to a next part in the logical data flow, the flow of record data during data processing being substantially continuous so as to form a stream of record processing from operator to operator within nodes and across nodes of the network.

2. A processor as claimed in Claim 1 wherein the record data in the stream of record processing may exist in various states at different parts in the data flow, and the parts in the logical data flow include on disk storage, within JPU

memory, on the network, within host computer memory, and within an ODBC connection with an end user or application.

3. A processor as claimed in Claim 1 wherein the plurality of operators includes a merge aggregation operator that determines record readiness based on a key index value, such that the merge aggregation operator aggregates a sorted record stream and outputs the aggregation associated with a current key index value whenever a new key index value is received as input.

4. A processor as claimed in Claim 1 wherein record readiness is determined by buffer status such that a communication layer sends a partial set of records across the network when its buffers are filled, without waiting for a working sequence of operators that produced the record data to complete before any records are sent across the network.

5. A processor as claimed in Claim 1 further comprising at least one programmable streaming data processor (PSDP) coupled to a respective JPU, the PSDP being one part in the logical data flow and processing data fields within records as buffers of records are received from a storage disk or an external network connection, without waiting to process any records until all records are received.

6. A processor as claimed in Claim 5 wherein the data fields are processed by the PSDP to produce virtual fields.

7. A processor as claimed in Claim 6 wherein the virtual fields are selected from a group consisting of: a row address, pad words (tuple scratch pad), a Boolean results from each of the filter operations, a hash result, a tuple null vector, a tuple length, and combinations thereof.

8. A processor as claimed in Claim 1 wherein each software operator follows a common data handling paradigm such that each operator can operate in any part of the logical data flow, the common data handling including each operator being able to accept one or more streams of record data as inputs and producing a stream of record data as an output.

9. A processor as claimed in Claim 8 wherein any operator may take as its input a stream of record data that is produced as the output of any other operator.

10. A processor as claimed in Claim 8 wherein certain ones of the operators materialize data and do so as sets of records.

11. A processor as claimed in Claim 8 wherein the operators further enable same algorithms to be used for a given operation whether that operation is executed on the host computers or on the JPUs.

12. A processor as claimed in Claim 1 wherein record data are processed at intermediate parts on the logical data flow as a collection of data field values in a manner free of being materialized as whole records between two successive operators.

13. A processor as claimed in Claim 12 wherein the plurality of operators includes one or more join operators, each join operator having multiple input streams and an output stream with references to original records in their packed form, and the output stream for the operator referring to data field values within the record data of the input streams at known offsets from a base pointer to a start of a packed record.

14. A processor as claimed in Claim 1 in which the JPU's CPU eliminates unnecessary data before it is sent across the network.

15. A processor as claimed in Claim 1 wherein at least one of the host computers eliminates unnecessary information before processing a next step of a subject query.

16. A processor as claimed in Claim 1 wherein the host computers further include a Plan Generator component, the Plan Generator component generating record data processing plans having operations which take input streams of record data and produce streams of record data as output and which avoid intermediate materialization.

17. A processor as claimed in Claim 1 wherein the host computers further include a Communication Layer API that accepts data records as input to a message sent to one or more other nodes.

18. A processor as claimed in Claim 1 wherein the host computers further include:
a Job Listener component for awaiting data from other nodes; and
an API which provides streams of record data as output.

19. A processor as claimed in Claim 18 wherein the host computers further comprise a Host Event Handler component for managing execution of a query execution plan, the Host Event Handler receiving partial result sets from JPUs through the Job Listener component.

20. A processor as claimed in Claim 1 wherein the host computers further comprise a Host Event Handler for managing execution of a query execution plan, the

Host Event Handler communicating to JPUs through a Communication Layer component to request partial result sets from JPUs.

21. A processor as claimed in Claim 20 wherein the Host Event Handler requests partial result sets from JPU buffers in order to get, sort and process partial result sets held in the JPU buffers instead of waiting for a JPU to fill its buffer and send the data to a host computer.

22. A processor as claimed in Claim 1 wherein the host computers include a Loader component which operates in streaming fashion and performs multiple operations on each field value in turn while each field value is held in a host CPU cache.

23. A processor as claimed in Claim 22 wherein the Loader component performs operations including one or more of: parsing, error checking, transformation, distribution key value calculation, and saving the field value to internal network output frame buffers.

24. A processor as claimed in Claim 1 wherein the JPUs separate the stream of record processing from source of the record data such that various input sources to the JPUs are permitted.

25. A processor as claimed in Claim 1 wherein the JPUs further comprise a Network Listener component which awaits requests from other nodes in the network and which returns a stream of record data as output.

26. A processor as claimed in Claim 1 wherein the JPUs further comprise a Network Poster component which accepts a stream of record data as input and which

sends data to other nodes when its buffers are filled, when jobs are completed or upon an explicit request to do so.

27. A processor as claimed in Claim 1 wherein the JPUs further comprise a Storage Manager component whose API and implementation provide for storage and retrieval of record sets.

28. A processor as claimed in Claim 1 wherein the host computers are of a symmetric multiprocessing arrangement and the JPUs are of a massively parallel processing arrangement.

29. A processor as claimed in Claim 1 wherein a node executes multiple operations on the subject record before processing a next record data.

30. A method of data processing comprising the steps of:

providing one or more host computers, each including a memory, a network interface and at least one CPU, each host computer being responsive to requests from end users and applications to process data;

providing one or more Job Processing Units (JPUs), each having a memory, a network interface, one or more storage devices, and at least one CPU, each JPU being responsive to requests from host computers and from other JPUs to process data;

networking the host computers and the JPUs to communicate between and amongst each other, each of the host computers and JPUs forming a respective node on the network; and

using a plurality of software operators, enabling each node to process data in a record- by -record, streaming fashion in which (i) for each operator in a given sequence of said operators, output of the operator is input to a respective succeeding operator in a manner free of necessarily materializing data, and (ii)

data processing follows a logical data path formed of node locations and operators and is based on readiness of a record, such that as soon as a subject record is ready, record data is passed from one node location or operator to a next node location or operator for processing along the logical data path, the flow of record data on the logical data path during data processing being substantially continuous so as to form a stream of record processing from operator to operator across nodes and within nodes of the network.

31. The method of Claim 30 wherein the record data in the stream of record processing may exist in various states at different node locations of the logical data path, and the node locations on the logical data path include on disk storage, within JPU memory, on the network, within host computer memory, and within an ODBC connection with an end user or application.

32. The method of Claim 30 wherein the plurality of operators includes a merge aggregation operator that determines record readiness based on a key index value, such that the merge aggregation operator aggregates a sorted record stream and outputs the aggregation associated with a current key index value whenever a new key index value is received as input.

33. The method of Claim 30 further comprising the step of determining record readiness as a function of buffer status such that a communication layer sends a partial set of records across the network when its buffers are filled, without waiting for a working sequence of operators that produced the records to complete before any records are sent across the network.

34. The method of Claim 30 further comprising the step of following a common data handling paradigm for each software operator such that each operator can operate at part of the logical data path, the common data handling including each

operator being able to accept one or more streams of record data as inputs and producing a stream of record data as an output.

35. The method of Claim 34 wherein any operator may take as its input a stream of record data that is produced as the output of any other operator.

36. The method of Claim 34 wherein certain ones of the operators materialize data and do so as sets of records.

37. The method of Claim 34 wherein the operators further enable same algorithms to be used for a given operation whether that operation is executed on the host computers or on the JPUs.

38. The method of Claim 30 further comprising the step of processing record data at intermediate locations on the logical data path as a collection of data field values, in a manner free of being materialized as whole records between two successive operators.

39. The method of Claim 38 wherein the plurality of operators includes one or more join operators, each join operator having multiple input streams and an output stream with references to original records in their packed form, and the output stream of the join operator referring to data field values within the record data of the input stream at known offsets from a base pointer to a start of a packed record.

40. The method of Claim 30 wherein the step of providing host computers includes generating record data processing plans formed of at least one sequence of operators from the plurality of operators, each sequence taking a stream of

record data on input and producing a stream of record data as output and avoiding intermediate materialization.

41. The method of Claim 30 further comprising the step of accepting data records as input to a message sent to one or more other nodes.

42. The method of Claim 30 further comprising the step of managing execution of a query execution plan including requesting partial result sets from JPU buffers in order to get, sort and process partial result sets held in the JPU buffers instead of waiting for a JPU to fill its buffer and send the data to a host computer.

43. The method of Claim 30 further comprising the step of performing multiple operations on each field value in turn while each field value is held in a host CPU cache.

44. The method of Claim 43 wherein the multiple operations include one or more of: parsing, error checking, transformation, distribution key value calculation, and saving the field value to internal network output frame buffers.

45. The method of Claim 30 further comprising the step of separating the stream of record processing from source of the record data such that various input sources to the JPU's are permitted.

46. The method of Claim 30 further comprising the step of sending data to nodes when a buffer is filled, when a job is completed or upon request.

47. The method of Claim 30 wherein the step of enabling includes at a node, executing multiple operations on the subject record before processing a next record data.